AF/2180#

TRANSMITTAL OF APPEAL BRIEF (Large Entity)

Docket No. 9295(NCR.0020US)

In Re Application Of: **WILLIAM P. WARD**

Serial No.	Filing Date	Examiner	Group Art Unit
09/651,229	08/25/2000	P.M. BATAILLE	2186

Invention: **PROVIDING MULTIPLE MEMORY CONTROLLERS ON A MEMORY BUS**

TO THE COMMISSIONER FOR PATENTS:

Transmitted herewith in triplicate is the Appeal Brief in this application, with respect to the Notice of Appeal filed on

The fee for filing this Appeal Brief is: **$320.00**

☐ A check in the amount of the fee is enclosed.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☒ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. **50-1673(9295)**

Dan C. Hu

Signature

Dated: 5-30-03

Dan C. Hu
Reg. No. 40,025
TROP, PRUNER & HU, P.C.
8554 Katy Freeway
Suite 100
Houston, Texas 77024

I certify that this document and fee is being deposited on **May 30, 2003** with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Dawn L. Thomas

Signature of Person Mailing Correspondence

Dawn L. Thomas

Typed or Printed Name of Person Mailing Correspondence

cc:

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

#13
JK 6-6-03

Applicant: William P. Ward	§	
Serial No.: 09/651,229	§	Group Art Unit: 2186
Filed: August 25, 2000	§	Examiner: P.M. Bataille
For: Providing Multiple Memory Controllers on a Memory Bus	§	Atty. Dkt. No. 9295 (NCR.0020US)

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Dear Sir:

Appellant hereby appeals from the Final Rejection dated December 24, 2002.

I. REAL PARTY IN INTEREST

The real party in interest is NCR Corporation by virtue of the Assignment recorded at Reel/Frame No. 011145/0568.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

III. STATUS OF THE CLAIMS

Claims 1-30 are finally rejected and are the subject of this appeal.

IV. STATUS OF AMENDMENTS

No amendments were made after final rejection.

Date of Deposit: May 30, 2003
I hereby certify under 37 CFR 1.8(a) that this correspondence is being deposited with the United States Postal Service as **first class mail** with sufficient postage on the date indicated above and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.
Dawn L. Thomas
Dawn L. Thomas

V. SUMMARY OF THE INVENTION

The following sets forth a discussion of some embodiments of the invention. Referring to Fig. 1, in accordance with one embodiment, a system 10 includes multiple memory controllers 12, 14, 16 and 18 that are connected to a memory bus 20. In one example, the memory bus 20 is a Rambus channel. However, other types of memory buses may be employed in further embodiments. Although not shown, the memory controllers 12, 14, 16, and 18 can be associated with one or more central processing units (CPUs). The memory controllers can be embedded in corresponding CPUs, or alternatively, the memory controllers can be separate from but associated with the CPUs. Specification, page 3, lines 17-23.

Optionally, the memory bus 20 is connected to a memory hub 22, which is in turn connected to one or more other memory buses 32. In one example, the one or more other memory buses 32 are also Rambus channels. Effectively, the memory hub is an expander buffer device. Specification, page 3, lines 24-27.

As shown in Fig. 1, a set of memory units 24 are connected to the memory bus 20. If optional memory bus 32 is present, then another set of memory units 26 (connected to the memory bus 32) is accessible by the memory controllers 12, 14, 16 and 18 through the hub 22. The memory units 24 and 26 in one example are Rambus® dynamic random access memories (RDRAMs). In one arrangement, up to 32 RDRAM chips can be connected to a single Rambus channel. Specification, page 3, line 28-page 4, line 2.

Each of the memory controllers 12, 14, 16, and 18 can generate a memory request on the bus 20 to access the memory units 24 (and optionally, units 26). A memory request includes control and address signaling. In addition, for a write cycle, a memory controller

provides write data on the memory bus 20. For a read cycle, an addressed memory unit returns the read data on the memory bus 20. On a Rambus channel, data and control/address signaling are communicated in packets, with each packet being four bus clock cycles in length. Data is transmitted on both the rising and falling edges of the clock. Conventional memory buses such as Rambus channels are usually associated with a single memory controller. As a result, tags or other types of identifiers are usually not provided with data on the memory bus to identify the destination memory controller. Conventionally, it is assumed that all data on the memory bus is associated with the single memory controller. There is also no priority selection mechanism for accessing the data bus because there are no conflicts in accessing the data bus. Thus, if multiple memory controllers are present, then a technique is needed to enable sharing of the memory bus 20 by the multiple memory controllers. Specification, page 4, lines 3-17.

In one arrangement, a time slot priority scheme is used in which the memory controllers 12, 14, 16 and 18 each generates a request within its assigned time slot. As used here, a "time slot" has a length equal to the length of a packet. Thus, in each time slot, a memory controller can communicate a packet (control or data). The time slot approach is advantageous in systems with more than one Rambus channel. In a system with equal numbers of independent memory controllers and Rambus channels, each memory controller would be enabled on one Rambus channel in each time slot. Most memory read operations are for cache blocks. The Rambus channel data packet delivers 16 bytes of data. This is half to one fourth the size of the cache block in many systems or devices. By interleaving the memory addressing, any single cache block can be spread across all the Rambus channels, so

the start of the read access for any processor request for a cache block can start on the next time slot if the memory is not busy. Specification, page 4, lines 18-29.

In another arrangement, instead of a time slot priority scheme, a request-select priority scheme can be used, in which memory controllers 12, 14, 16, and 18 can assert request lines to gain access to the bus 20. In this case, an arbiter (not shown) can be provided on the bus 20 to arbitrate between requests from the memory controllers. Alternatively, one of the memory controllers or all of the memory controllers 12, 14, 16, and 18 can be designated as the arbiter. Since a memory bus has separate row and column buses, a separate request selection scheme may be employed for each of the row and column buses. Alternatively, request selection may be performed on the row bus, with a predetermined delay set for access to the column bus. Further, data on the data bus also occurs a predetermined time after a read or write command has been communicated on the column bus. Specification, page 4, lines 31-32, Specification, page 5, lines 1-9.

To implement the time slot priority scheme for the four memory controllers 12, 14, 16, and 18, four control clocks (represented as blocks 60, 62, 64 and 66) are provided to the corresponding four memory controllers . The active states of the control clocks 60, 62, 64 and 66 are out of phase so that each processor is enabled for access to the memory bus 20 during its assigned time slot. The control clocks 60, 62, 64 and 66 are synchronized to the memory bus clock (represented as block 68). Although separate blocks are used to represent the clocks 60, 62, 64, 66, and 68, the clocks can all be generated by the same clock generator. Specification, page 5, lines 10-18.

Control and data information from the memory controllers 12, 14, 16, and 18 are transferred to the memory bus 20. The addressed one of the memory units 24 responds to the

requesting memory controller. A predetermined time delay is provided between a request from the memory controller and data returning from the addressed memory unit. Because of this latency, the memory controllers 12, 14, 16 and 18 can time their requests so that data from the memory units come back in different time slots to the memory controllers. Thus, for example, the memory controller 12 can issue a first request, followed by the memory controllers 14, 16 and 18, successively. The multiple memory controllers 12, 14, 16, and 18 can have multiple memory requests outstanding on the memory bus 20. Each memory controller 12 tracks memory requests from other memory controllers. Specification, page 5, lines 19-29.

As shown in Fig. 2, the memory bus 20 (implemented as a Rambus channel in the illustrated embodiment) is separated into several major portions: a data bus 200; a row command/address bus 202 (hereinafter "row bus"); a column command/address bus 204 (hereinafter "column bus"); Rambus channel control signals 206; and special signals 208 provided to enable lock and cache control between the memory controllers 12, 14, 16 and 18. Further, the special signals 208 include request/select signal(s) if a request/select priority scheme is used. Specification, page 5, lines 30-31, Specification, page 6, lines 1-5.

Communication of the various signals are handled by memory control portions 100, 102, 104, and 106 in respective memory controllers 12, 14, 16, and 18. Cache control portions 110, 112, 114 and 116 may also be involved for cache coherency purposes. The cache control portions 110, 112, 114, and 116 manage associated caches, which can be level-one (L1) or level-two (L2) caches. Specification, page 6, lines 6-10.

In one arrangement of the Rambus channel, the data bus is a 16-bit data bus containing signals DQA[8:0] and DQB[8:0]. The row control/address bus 202 includes ROW[2:0]

signals and the column control/address bus 204 includes COL[4:0] signals. The row, column and data buses can operate independently of each other. This allows commands, addresses, and data to be communicated concurrently to different banks of RDRAMs or from different RDRAM devices. The control signals include a clock-to-master signal ClkToMaster (and its complement) and a clock-from-master signal ClkFromMaster (and its complement). The voltages include various supply, reference, and ground voltages. Also, the control signals 206 include signals used for initialization (not shown). Specification, page 6, lines 11-20.

Row packets are sent across the three ROW[2:0] pins of the row bus. Row packets include two types of commands: activate (ACT) or precharge (PRER). Column packets are sent across the five COL[4:0] pins of the column bus. The COL packets are split into two fields. The first field specifies the primary operation, such as a read or write. The second field can be either masks, for writes, or an extended operation command. Each data packet communicated over the DQA[8:0] and DQB[8:0] pins contains 16 bytes of data. Specification, page 6, lines 21-27.

A read operation is generally performed by asserting an ACT command across the ROW pins followed by a read command across the COL pins. Read data is returned in a predetermined number of bus cycles from the end of the read command. This read delay can be varied in the system to accommodate different bus arrangements. The timing of write transaction commands is similar to those of read transactions, with a write command sent in the same way as a read command. The write data for a write transaction is delayed to match the timing of a data transfer of a read transaction. Specification, page 6, line 28-page 7, line 3.

The special signals 208 include a LOCK/UNLOCK signal that can be used to enable read-modify-write cycles. In one embodiment, separate LOCK and UNLOCK signals are used. In another embodiment, the LOCK/UNLOCK signal is a single signal in which a low or high state indicates whether a lock is in place. The purpose of asserting a lock is to prevent a second memory controller from modifying a memory location after it has been read by the first memory controller but before the first memory controller has been written back to memory. Specification, page 7, lines 4-10.

The special signals 208 also include an ABORT signal to enable a cache control portion in a first memory controller to abort a read or write access from another memory controller if the cache associated with the first memory controller contains modified data. Also, one or more REQUEST/SELECT signals can be part of the special signals 208 to enable a request-select priority scheme. If a time slot priority scheme is used, however, then the REQUEST/SELECT signals are not needed. Specification, page 7, lines 11-16.

Referring to Fig. 3, the time slot priority scheme on the row and column buses 202 and 204 is illustrated. Each illustrated box represents a time slot on the corresponding bus. Thus, for example, for a four-memory controller system, four time slots can be defined in which a different memory controller is able to communicate a packet during that time slot. Thus, as shown in Fig. 3, a first time slot 210 on the row bus 202 is allocated to memory controller 1, a second time slot 212 is allocated to memory controller 2, a third time slot 214 is allocated to memory controller 3, and a fourth time slot 216 is allocated to memory controller 4. This pattern is then repeated in round robin fashion. Similarly, on the column bus 204, a first time slot 220 is allocated to memory controller 4, a second time slot 222 is allocated to memory controller 1, a third time slot 224 is allocated to memory controller 2, and a fourth time slot

226 is allocated to memory controller 3. The staggering between the row and column buses 202 and 204 is provided since the memory controller typically asserts a row command/address packet followed by a column command/address packet. Specification, page 7, lines 17-30.

To control when a memory controller is active on the memory bus, the control clocks 60, 62, 64 and 66 (Fig. 1) are used. As noted above, the active states of the control clocks 60, 62, 64 and 66 are phased to provide the time slot arrangement. The memory controller is enabled to generate memory requests when its associated control clock is active. For the independent row and column buses, the control clock 60, 62, 64 or 68 can be used to derive the time slot arrangement for each of the row and column buses. Each memory controller can have an internal delay mechanism that is based on the control clock 60, 62, 64 or 68 to determine when it is active on the row bus and column bus. Specification, page 8, lines 1-8.

By enabling the connection of multiple memory controllers to a single bus (or to plural buses), flexibility is provided in how the system 10 can be arranged. For example, the system 10 can be a multiprocessor system, with each memory controller associated with a respective CPU. The system 10 can also be a uniprocessor system, with the plural memory controllers provided to enhance memory access speeds and to increase bandwidth. Specification, page 8, lines 9-14.

Referring to Fig. 4, a timing diagram for a series of read transactions is shown for two memory controllers (memory controller 1 and memory controller 2). The example can also be extended for more than two memory controllers. In time slot 300, memory controller 1 places a first row command packet on the row bus 202. In the next time slot 302, memory controller 1 places the column command packet (read command) on the column bus 204. In the same time slot 302, memory controller 2 places its first row command packet on the row bus 202,

since the row bus is not being used by memory controller 1 during this time slot. In the next time slot 304, memory controller 1 places its second row command packet on the row bus 202 and memory controller 2 places its first column command packet on the column bus 304. In time slot 306, memory controller 1 provides its second column command packet on the column bus 202, and memory controller 2 places its second row command packet on the row bus 202. In the same time slot 306, read data is transmitted by an addressed memory unit on the data bus 200, the read data being the data requested by memory controller 1 in time slots 300 and 302. In the next time slot 308, read data requested by memory controller 2 is returned on the data bus 200. In the same time slot 308, memory controller 1 places its third row command packet on the row bus 202, while memory controller 2 places its second column command packet on the column bus 204. Successive row and column packets along with read data are communicated in subsequent time slots 310, 312, 314, 316, 318, 320 and 322 until all requested data have been returned to memory controllers 1 and 2. Specification, page 8, line 15-page 9, line 3.

As illustrated, the two memory controllers are able to interleave the row and column command packets so that use of the row and column buses is optimized. Specification, page 9, lines 4-5.

Referring to Fig. 5, a timing diagram for a series of read and write transactions is illustrated. In a first time slot 400, memory controller 1 places its first row command packet on the row bus 202. In the next time slot 402, memory controller 1 places its first column command packet (read command) on the column bus 402, while memory controller 2 places its first row command packet on the row bus 202. In the next time slot 404, memory controller 2 places its first column command packet (write command) on the column bus 204.

However, in this time slot, memory controller 1 does not drive either the row or column buses since a wait cycle is needed between writes and reads on the data bus 200. Specification, page 9, lines 6-14.

In the next time slot 406, memory controller 2 places its second row command packet on the row bus 202. In the same time slot 406, the read data for memory controller 1 is returned on the data bus 200. Specification, page 9, lines 15-17.

In the next time slot 408, memory controller 1 places its second row command packet on the row bus 202, and memory controller 2 places its second column command packet (read command) on the column bus 204. In the same time 408, memory controller 2 places write data on the data bus 200 for writing to the addressed memory unit. As illustrated, the delay between the write command and the write data is set to match the delay between a read command and read data. Specification, page 9, lines 18-23.

In the next time slot 410, memory controller 1 places its second column command packet (read command) on the column bus 204, and memory controller 2 places its third row command packet on the row bus 202. However, the data bus is idle in this time slot to provide the wait period between write data and read data. Specification, page 9, lines 24-27.

In the next time slot 412, memory controller 1 places its third row command packet on the row bus, and memory controller 2 places its third column command packet (read command) on the column bus. Also, read data for memory controller 2 is returned on the data bus 200 in time slot 412. A series of subsequent reads are then performed in time slots 414, 416, 418, 420 and 422. Specification, page 9, line 28-page 10, line 2.

The memory controllers monitor command packets (control and address) communicated by other memory controllers to determine what memory transactions are

occurring. By monitoring what other memory controllers are doing, each controller can avoid issuing a request that may conflict with another memory controller. Also, by monitoring the memory requests, lock and cache control may also be performed to maintain data integrity. Specification, page 10, lines 3-8.

Referring to Fig. 6, a memory controller monitors (at 702) transactions on the memory bus. Such monitoring involves monitoring the commands and addresses communicated in packets issued by other memory controllers. When the memory controller receives (at 704) a request to generate a memory cycle (read, write, refresh, etc.), it determines when the data bus will become available (at 706). This is performed by determining what requests are outstanding, and when the data buses may be occupied or otherwise be unavailable. For example, a write transaction may have been performed by another memory controller that would require a wait cycle to be inserted on the data bus. In this instance, the memory controller desiring to issue a memory request may skip one time slot to prevent a memory conflict on the data bus. Thus, if necessary, the memory controller waits (at 708) to generate the desired memory request. Specification, page 10, lines 9-19.

During initialization, one or more of the memory controllers is assigned the task of making all of the memory refresh requests to memory. Specification, page 10, lines 20-21.

Interaction between the multiple memory controllers is also needed to handle read-modify-write transactions. A read-modify-write transaction involves reading data from an address location in a memory device, modifying the data in the memory controller (or other device such as a CPU), and writing the modified data back to the same address location. During the time between the read and write steps, access (read or write) by other memory controllers to the same address location should be prevented. Referring to Fig. 7, to perform a

read-modify-write transaction the memory controller issues (at 750) the necessary row and column commands to perform a read transaction. Next, the memory controller asserts a LOCK signal (at 752) to lock the accessed memory location. The memory controller receives the read data (at 754) and modifies the read data (at 756). The memory controller then issues a write command (at 758) back to the same address location. Some predetermined time later, the memory controller sends the write data (at 760) to the memory unit. When the write cycle is completed, the memory controller asserts the UNLOCK signal (at 762) to enable other memory controllers to access the desired data. Specification, page 10, line 22-page 11, line 5.

Referring to Fig. 8, when a memory controller receives an indication (such as from an associated CPU) to generate a read or write request (at 770), the memory controller first checks (at 772) to determine if a lock has been asserted for the requested memory location. If so, the memory controller waits (at 774) until the lock has been released. If a lock is not asserted, the memory controller proceeds (at 776) with the read or write request. Specification, page 11, lines 6-11.

Referring to Fig. 9, acts performed by the memory controller (involving the memory control and cache control portions) for cache coherency is illustrated. The memory controller first determines if an address snoop hit event has been received (at 802), which occurs when the memory controller detects a memory address on the row and column buses that matches the memory address of a cache line stored in the associated cache. Specification, page 11, lines 7-17.

When a snoop hit event is received, the memory controller determines (at 804) if the memory request is a read or write. If the request is a write, then the memory controller sets (at 806) the state of the cache line to invalid. At some later point, the memory controller may

issue read requests to update (at 808) the cache line with the latest data. After the update, the cache line is set (at 810) to the valid state. Specification, page 11, lines 18-22.

If the bus transaction that caused the snoop hit is a read transaction, the memory controller determines (at 812) if the cache line corresponding to the address of the read transaction contains dirty (or modified) data. If the cache line contains dirty data, then the memory controller asserts (at 814) the ABORT signal, which is received by the requesting memory controller. After asserting the ABORT signal, the memory controller issues a write command (at 816) to write the updated cache line back to the memory subsystem. As the cache line is being written back to the memory subsystem, the other memory controller will see the write data and update its cache accordingly. Specification, page 11, lines 23-30.

If the memory controller determines (at 812) that the cache line does not contain modified data, the memory controller then determines (at 818) if the read transaction is a private or shared read. A private read is typically performed by a device wishing to modify the data. If the read is determined to be a shared read (at 818), then no further cache action is needed. However, if the read is a private read, then the memory controller sets (at 820) the cache line to the invalid status. Specification, page 12, lines 1-6.

Although some embodiments of the invention are described above, it is noted that other embodiments are within the scope of the claims on appeal.

VI. ISSUES

A. **Can A Reference That Does Not Teach Or Suggest Recited Elements Of Claims 10-23 and 26-30 Anticipate Or Render Obvious Those Claims?**

B. **Can A Reference That Does Not Disclose Recited Elements Of Claims 1-3, 5, 9, 24, and 25 Anticipate Those Claims?**

C. **Can A Reference That Does Not Disclose Recited Elements Of Claim 4 Anticipate That Claim?**

D. **Can A Reference That Does Not Disclose Elements Of Claims 6 and 7 Anticipate Those Claims?**

VII. GROUPING OF THE CLAIMS

Group 1: Claims 10-23 and 26-30.

Group 2: Claims 1-3, 5, 9, 24, and 25.

Group 3: Claim 4.

Group 4: Claims 6 and 7.

Within each group, the claims stand and fall together.

VIII. ARGUMENT

All claims should be allowed over the cited references for the reasons set forth below.

A. **Can A Reference That Does Not Teach Or Suggest Recited Elements Of Claims 10-23 and 26-30 Anticipate Or Render Obvious Those Claims?**

Claim 10 was rejected as being anticipated by Welker.

Claim 10 recites a system having a memory bus and a plurality of memory controllers connected to *the* memory bus, with each memory controller to monitor memory requests generated by another memory controller in performing memory-related actions.

The Examiner asserted that the element "each memory controller to monitor memory requests generated by another memory controller" is satisfied by the teaching in Welker that "each MIC controller includes its own snoop controller 706 generating snoop cycles and

return snoop transactions response which the requesting controller acknowledges." 12/24/02 Office Action at 8. Appellant respectfully disagrees. Each MIC 310 in Welker does not monitor memory requests of another MIC 310. The snoop controller 706 within each MIC controller does not monitor memory requests generated by another MIC 310. The snoop controller 706 responds to a request from a bus master (212-220) by generating a snoop request to the central snoop arbiter 312 (Figure 3 of Welker). The central snoop arbiter 312 can receive multiple snoop requests from the MICs 310. The central snoop arbiter 312 selects one of these snoop requests and forwards it to the processor interface controller 300 (Figure 3) to perform the snoop cycle. Thus, it is clear that the snoop controller within each MIC does not monitor memory requests generated by another MIC. Rather, the snoop controller 706 forwards a snoop request to the central snoop arbiter 312. Any response regarding this snoop request is received back from the central snoop arbiter 312, which is not part of any MIC 310. *See* Welker, 9:7-20.

Therefore, independent claim 10 is allowable over Welker. Independent claims 15 and 23 are allowable over Welker for reasons similar to those given for claim 10.

Claim 23 was actually rejected as being obvious over Welker. However, this rejection is based on the Examiner's erroneous reading of Welker as disclosing one memory controller monitoring memory requests generated by another memory controller. Therefore, the Examiner has failed to establish a *prima facie* case that claim 23 is obvious over Welker.

Another point of distinction with respect to claims 10, 15, and 30 is that Welker does not disclose plural memory controllers on *a* memory bus. This point is discussed below in Sub-Section B.

For the foregoing reason, it is respectfully requested that the final rejection of claims 10-23 and 26-30 be reversed.

B. Can A Reference That Does Not Disclose Recited Elements Of Claims 1-3, 5, 9, 24, and 25 Anticipate Those Claims?

Claim 1 was also rejected as being anticipated by Welker.

Claim 1 recites a system having a memory bus, and a plurality of memory controllers, with each memory controller to generate memory requests on *the* memory bus according to a predetermined priority scheme. At least two of the plurality of memory controllers are adapted to generate concurrently pending memory requests on the memory bus.

Claim 1 is clearly allowable over Welker because it does not disclose plural memory controllers on *a* memory bus. In the Final Office Action, the Examiner pointed to the collection of the multi-channel memory interface 106 and Rambus channels 202-208 of Welker as disclosing the memory bus. 12/24/02 Office Action at 5. The objective evidence establishes, however, that it is not the collection of Rambus channels 202-208 that forms the memory bus--more accurately, one Rambus channel is considered one memory bus. *See* Welker, 3:37 ("The Rambus channel is a synchronous high speed *bus* . . . .") (emphasis added).

After Appellant pointed out this defect in the Examiner's rejection, the Examiner argued in an Advisory Action dated March 4, 2003 that Welker discloses that each MIC 310 is coupled to the bus masters 212-220 through a single address, data, and control bus 314. Note that this argument was raised for the first time after the final rejection, and thus constitutes a new ground of rejection at this stage, which is improper. The bus 314 *cannot* be the memory bus recited in claim 1, because the MICs 310 do *not* generate memory requests

on the bus 314. The MICs 310 receive requests from bus masters over the bus 314--the MICs do not generate memory requests on the bus 314.

Thus, Welker fails to teach or suggest that a plurality of memory controllers are able to generate memory requests on a memory bus, where at least two of the memory controllers are adapted to generate concurrently pending memory requests on *the* memory bus. As shown in Figure 3 of Welker, only one memory interface control block (MIC) 310 is connected to one Rambus channel. There is no indication that two or more MICs 310 can generate requests on one Rambus channel.

The memory interface 200 of Welker includes four memory interface controllers (MICs) 310, as shown in Figure 3. A key teaching in Welker is that "[e]ach MIC 310 *independently* controls access to its respective memory channel 202-208." Welker, 7:28-31 (emphasis added). In other words, in Welker, one memory controller independently controls access to one memory bus.

This teaching is contrasted to what is recited in claim 1, namely a system that comprises a memory bus and a plurality of memory controllers, with each memory controller to generate memory requests on *the* memory bus. Claim 1 is explicitly clear that each of the plurality of the memory controllers is able to generate memory requests on the *same* memory bus. In fact, claim 1 further recites that at least two of the plurality of memory controllers are adapted to generate concurrently pending memory requests on *the* memory bus.

In contrast, Welker recites four separate and *independent* buses (the Rambus channels) and four separate and *independent* memory controllers to generate memory requests on each respective memory bus. Two MICs 310 as disclosed in Welker are unable to generate memory requests on the same memory bus.

Therefore, Appellant respectfully requests that the final rejections of claims 1-3, 5, 9, 24, and 25 be reversed.

C. Can A Reference That Does Not Disclose Recited Elements Of Claim 4 Anticipate That Claim?

Claim 4 depends from claim 1 and is thus allowable for at least the same reasons. Moreover, claim 4 further recites that the memory bus comprises a Rambus channel. In other words, in claim 4, the plurality of memory controllers are able to generate memory requests on the same Rambus channel. This clearly cannot be achieved and is not disclosed by Welker. Appellant respectfully requests that the final rejection of claim 4 be reversed.

D. Can A Reference That Does Not Disclose Elements Of Claims 6 and 7 Anticipate Those Claims?

Dependent claim 6 depends from claim 1 and is allowable over Welker for at least the same reasons as claim 1. Moreover, claim 6 further recites that the memory bus comprises plural control portions, with each of the control portions associated with corresponding time slot priority schemes. The Office Action pointed to the MICs 310 as being the recited plural control portions. The MICs 310 were cited in the Office Action as being the memory controllers with respect to claim 1. The MICs 310 do not constitute control portions of a memory bus. Examples of control portions of a bus are control signals on the bus. Welker does not disclose the further recited features of claim 6.

Therefore, Appellant respectfully requests that the final rejections of claims 6 and 7 be reversed.

VIII. CONCLUSION

It is respectfully submitted that each of the final rejections be reversed and that the claims subject to this appeal be allowed to issue.

Respectfully submitted,

Date: 5-30-03

Dan C. Hu, Reg. No. 40,025
Trop, Pruner & Hu, P.C.
(713) 468-8880 [Phone]
(713) 468-8883 [Fax]

APPENDIX OF CLAIMS

The claims on appeal are:

1. A system comprising:

a memory bus; and

a plurality of memory controllers, each memory controller to generate memory requests on the memory bus according to a predetermined priority scheme,

at least two of the plurality of memory controllers adapted to generate concurrently pending memory requests on the memory bus.

2. The system of claim 1, wherein the predetermined priority scheme comprises a time slot priority scheme.

3. The system of claim 1, wherein the predetermined priority scheme comprises a request-select priority scheme.

4. The system of claim 1, wherein the memory bus comprises a Rambus channel.

5. The system of claim 1, wherein each memory controller generates a memory request during a different predetermined time slot.

6. The system of claim 1, wherein the memory bus comprises plural control portions, each of the control portions associated with corresponding time slot priority schemes.

7. The system of claim 6, wherein the time slot priority schemes are staggered.

8. The system of claim 6, wherein the control portion comprise a row portion and a column portion.

9. The system of claim 1, wherein the memory bus comprises plural portions, each portion associated with a set of memory devices.

10. A system comprising:

a memory bus; and

a plurality of memory controllers connected to the memory bus, each memory controller to monitor memory requests generated by another memory controller in performing memory-related actions.

11. The system of claim 10, wherein the memory-related actions comprise a read-modify-write transaction.

12. The system of claim 10, wherein the memory-related actions comprise a cache coherency action.

13. The system of claim 10, wherein the memory-related actions comprise a memory request.

14. The system of claim 10, the memory controller to determine if the memory bus is available based on outstanding requests from other memory controllers.

15. A method comprising:

providing multiple memory controllers on a memory bus;

generating requests, by the memory controllers, on the memory bus; and

each memory controller monitoring memory-related actions by at least another memory controller.

16. The method of claim 15, wherein generating the requests comprises generating Rambus command packets.

17. The method of claim 15, wherein generating the requests comprises the memory controllers generating the requests one at a time according to a predetermined priority scheme.

18. The method of claim 17, wherein generating the requests comprises generating the requests according to a time slot priority scheme.

19. The method of claim 17, wherein generating the requests comprises generating the requests according to a request-select priority scheme.

20. The method of claim 15, further comprising each memory controller determining when to generate a memory request based on the monitoring.

21. The method of claim 15, further comprising each memory controller determining if a lock has been asserted due to presence of a read-modify-write transaction.

22. The method of claim 15, further comprising each memory controller performing a cache coherency action based on the monitoring.

23. An article comprising one or more storage media containing instructions that when executed cause a memory controller to:

monitor memory requests from another memory controller on a memory bus;

determining if a memory request can be generated on the memory bus based on the monitoring.

24. The system of claim 1, wherein the plurality of memory controllers are connected to the memory bus.

25. The system of claim 1, wherein one of the at least two memory controllers is adapted to generate its memory request on the memory bus before data is returned for the memory request of the other one of the at least two memory controllers.

26. The system of claim 10, wherein at least two of the memory controllers are adapted to generate concurrently pending memory requests on the memory bus.

27. The system of claim 26, wherein one of the at least two memory controllers is adapted to generate its memory request on the memory bus before data is returned for the memory request of the other one of the at least two memory controllers.

28. The method of claim 15, wherein generating the requests on the memory bus comprises at least two of the memory controllers generating concurrently pending requests on the memory bus.

29. The method of claim 28, wherein generating concurrently pending requests comprises one of the at least two memory controllers generating its request before data is returned for the request of the other of the at least two memory controllers.

30. The article of claim 23, wherein the memory controllers are connected to the memory bus.